September/October 19[illegible] Vol. 4 Nu[illegible] $3.00 (North America)

the work[illegible] micros

Support for Motorola based computer systems and microcontrollers

The Last Atlanta CoCoFest!... but the first Picnic!

It is my sad duty to report of the demise of the Atlanta CoCoFest! With support both within the club and from the CoCo community diminishing each year, we feel that we can no longer organize an effective Fest. We can however organize a club picnic!

The Atlanta Computer Society, Inc. will be holding a picnic in Stone Mountain Park on October 5, 1996. This date corresponds to the date we would have had for a CoCoFest. This club picnic will be more than an ordinary picnic because we are hereby inviting any and all CoCo enthusiast, both attendees of past fests and vendors. We want to get together to celebrate our mutual interest in theCoCo.

The picnic will be held in the Virginia Pavilion on the ground of Stone Mountain Park. The Virginia Pavilion is located adjacent to the Antibellum Plantation and easy walking distance to the Stone Mountain Park Inn and the Laser show. There is an excellent camp ground within the park as well as two motels, the Evergreen and the Stone Mountain Park Inn, with excellent meals being served in both places.

Food and entertainment for all picnic attendees will be provided by ACS *FREE OF CHARGE*. Entrance fees to the park will be reimbursed to all club members of record on the day of the picnic. Non-members will have to pay a $6.00 daily entrance fee to enter the park (that's $6.00/vehicle, not per person). There will be games and prizes for club members and each club member will be guaranteed to get a prize.

Non-members are asked to confirm, in writing, that they will attend. Confirmations must be accompanied with a $1.00/person refundable binder. Attendees will be refunded that $1.00, it's just a method of counting noses so that we can prepare sufficient food for all attendees. Confirmations must be received by September 15 to be accepted. The picnic will start at 10:00 AM and end soon after the end of the Laser and fireworks show (approximatly 10:15 PM). Please plan on staying for that show, it will be worth your time. As an added attraction, Stone Mountain will be holding the "Great Miller Chili Cookoff", a fine place to taste some of the south's greatest Chili! Because of the Chili Cookoff it is suggested that you make camping or hotel reservations early. If you plan on attending please confirm, in writing, to: **Atlanta Computer Society, Inc., PO Box 80694, Atlanta, Ga. 30366.** Be sure to include $1.00 per person for each attendee. ***Vendors may sale items, but no power will be available for computer systems.***

Alan Dagoo. President, Atlanta Computer Society, Inc.

CONTENTS

Stone Mountain Park Inn rooms range from $59.00 to $79.00 and higher. Stone Mountain Park Inn can be reached at (800)277-0007. The Evergreen can be reached at (800)722-1000. The Campground does not normally take reservations but rest assured there are plenty of site available. Full hookups are $17.00. For general information about Georgia's Stone Mountain Park call (770)498-5690 or visit their web site at: http://www.stonemountainpark.org

POSTMASTER:
If undeliverable return to:
FARNA Systems PB
Box 321
WR, GA 31099

If your address is incorrect, send me a postcard!

the world of 68' micros

Publisher:
FARNA Systems PB
P.O. Box 321
Warner Robins, GA 31099-0321

Editor:
Francis (Frank) G. Swygert

Subscriptions:
US/Mexico: $24 per year
Canada: $30 per year
Overseas: $50 per year (airmail)
Back and single issues are cover price. Canada add $0.50 per issue ($1.00 max) additional shipping, overseas add $2.00 per issue. Bulk/newsstand orders available.

microdisk:
Companion to magazine. Contains program listings, shareware, and freeware as mentioned in magazine text. Three issues per year.
US/Mexico: $20 per year ($8 single)
Canada: $25 per year ($10 single)
Overseas: $40 per year ($15 single)

Advertising Rates:
Contact publisher. We have scales to suit every type of business. Special rates for entrepreneurs and "cottage" businesses.

Contributions:
Any and all contributions welcome. Submission constitutes warranty on part of the author that the work is original unless otherwise specified. Publisher reserves the right to edit or reject material without explanation. Editing will be limited to corrections and fitting available space. Authors retain copyright. Submission gives publisher first publication rights and right to reprint in any form with credit given author.

General Information:
Current publication frequency is bi-monthly. Frequency and prices subject to change without notice. All opinions expressed herein are those of the individual authors, not necessarily of the publisher. No warranty as to the suitability or operation of any software or hardware modifications is given nor implied under any circumstances. Use of any information in this publication is entirely at the discretion and responsibility of the reader.

All trademarks/names property of their respective owners

The publisher is available via e-mail at: dsrtfox@delphi.com

A message from the editor...

Here we are again, another issue already, and I have no idea what I'm going to write to fill this column! I'll have to just think real hard and see where we (you, me, the magazine, OS-9, the CoCo community... all of us!) go from here..

For starters, thanks to those who expressed an opinion as to inviting another computer to join our little community. Most people don't seem to mind at all, as long as they will continue to get good service. The general opinion is that it should be another "orphan" model, and not one that will take over.

With this in mind, I have decided to go ahead and make some contacts with the Atari users on the Internet and Delphi. I don't know what the outcome of this will be yet. If they agree to help promote the magazine and write some articles, you'll start to see some Atari additions around the first of the year, possibly one in the next issue. If I do indeed start to pick up a substantial number of subscriptions, you will continue to see Atari related articles. It is my hope that some of the hard and soft ware articles can cross platforms. If I don't get a decent number of Atari interested subscribers, I'll simply drop the articles and refund the few the remainder of their subscription price. The only thing I can do is give it a try.

There is no need to worry about the Atarians taking over or dominating the magazine. For one, I don't intend to add another computer to my collection. Someone else will be handling all the Atari software tests and such... will probably have an assisting editor to handle all the Atari input. I have the final say in the magazine, as I put it together and do all final editing. If the support is there, I don't mind having equal space for the Ataris, but they won't dominate. At the worst, Atari articles may dominate an issue or two, just like some issues are DECB dominated, some OS-9.

Ed Gresick is still hanging in there. Don't forget to send him a card to support him and his family in his hour of need. The Lord knows he has supported many OS-9 users throughout the years. The address is: Delmar Co. Box 78, Middletown, DE 19709-0078 For those who don't know, Ed has terminal cancer and will not be with us much longer.

One final item... the bell has tolled for the Atlanta CoCoFest. Attendance was okay last year, but just so. Since a smaller crowd was expected for this year, no one wanted to devote the time and effort required to make a really nice fest. You must remember that several members had to devote several days of their time to make sure the fest went smooth for everyone. Some gave a lot more time than others. Doing all the running around and arrangement making takes a LOT of time. Anyone who has helped organize an event of any size knows what I mean.

Fortunately, the Atlanta Computer Society decided that the fest shouldn't just die off. This year, they are organizing a picnic. There won't be regular vendor tables and such, but a couple vendors (such as myself) will be there with a few things to sell. I don't know if I'll be selling from my briefcase, a table, or out the trunk of the car, but I'll have a decent selection of "stuff". I'm sure a couple of other vendors will be doing the same.

If you noticed this issue was a little early, it was no accident. Hmmm.... I guess I really shouldn't say that. I got smart and started this issue real early, since Allen Huffman and a couple others sent in articles early (Allen's was broken into two parts...). When I got the latest info from ACS, I realized I needed to get it out to readers as soon as possible, so I jumped right to work finishing up... and here it is! Don't forget to read the front cover, and get your reservation in the mail! My wife and I plan on staying at the Stone Mountain Inn. Hope to see you there!

That about wraps it up this month. Hope you enjoy this issue. It is a couple pages shorter than the last, but there is good, informative content nontheless.

Messages from our readers...

Dear Frank:

I am looking for two DWP 230 printers and two DMP 430 and/or two DMP 420 printers at the market price.

By way of eaplanation, and to answer some of the questions your readers have about "don't know how long CoCo's are going to be in use" - my experience is as follows:

We use up to ten CoCo's running almost 24 hours a day. We are CPAs and I wrote most of the software to do payrolls, compilations, and reports. We print payroll checks and vendor payment checks and post all transactions to disk for further trial balance processing. We started using CoCos in 1981 and still use the original gray ones, but upgraded to CoCo2's and then CoCo3's and have a mix currently runnning. We had some problems with the CoCo3 CPU's at first, but got them repaired before all the repair centers stopped fixing CoCos. The CoCo's are all still alive and well. The printers have a very physical job, they tend to wear out faster. We intend to continue using the CoCos for another 10 years (we stocked up on CoCo2's when they were discontinued) as payroll processors and for data entry. The weak link is the printers.

We experimented with serial to parallel adapters, but the smaller dot matrix printers wouldn't do the hefty job of the DMP or heavy DMP printers we have been using. At this time I really would like to see if anyone has connected a laser printer to their CoCo so we can use laser checks. The serial to parallel adapters don't seem to do this.

Your magazine is really great. I don't get into the mechanics so much as I do the end-user utility of the CoCo. I can emphatically state that the CoCo processor (albeit slow in these times) works great and the slow speeds are compensated for by running up to 10 at a time for differing reports. I turn on the programs and go home for the evening. When I return in the morning they are all done. Besides, even though we will eventually go to MS-DOS machines (I am currently re-writing my code in GW BASIC), I can't run more than one printer per machine simultaneously even under Win95.

Ken Greiner, CPS
20 west Williams Street
Delaware, OH 43015

Wow Ken! That is quite an operation you have there! As long as the CoCos do their job, I'd keep them running too. Converting all that code to GW-BASIC is quite a chore, though you can then run it on a bunch of 286 or 386 machines when the CoCos finally get to slow (don't know if you want that much work to do or not!) or start becoming unreliable... something that will happen sooner or later under the long running hours you suggest.

You CAN use a laser printer with your CoCo. The problem may be that you are using DMP and DWP specific print codes. The newer lasers usually recognize that standard Epson/IBM codes, but not the older Tandy control codes. Get any Epson printer manual and it will have the correct codes in it. Make sure the laser printer you get emulates an Epson or IBM printer. Another option would be Diablo emulation. I know the Okidata 800 series lasers emulate a Diablo and an Epson printer. I mention the Diablo because that is the industry standard for Daisy Wheel printers (DWP), and the RS DWP may use those codes. Check surplus stores for original HP LaserJets. They don't work well with modern computers and are therefore pretty cheap. The CoCo should print to them and use the built-in font. You can get these with a standard 25 pin serial port... a cable can be easily made to use them.

I hope a few readers have those printers for you. Check local PC companies that do a lot of business upgrades also. Businesses used to use these, and a lot of similar printers (especially Diablos) are out there on the surplus market. These heavy duty printers are what you need, and dot matrix line printers.

One source might be Javanco (501 12th Avenue South, Nashville, TN 37203; phone 1-800-528-2626; BBS 615-244-4445; Fax 615-244-4446; http://www.javanco.com). They have lots of surplus items, including serial terminals for OS-9 users really cheap.

I just received my first issue of "the world of 68 micros" and I must say that I am quite happy with it. Regarding your editorial, have you considered bringing in the Atari ST, Amiga, and even MacIntosh crowds? I'm not talking about general or advocacy either, I mean real techinal articles dealing with realwork applications. That's just my $0.02 though. Hope things continue to go well.

Jim Cox
jimc@amc.com

Enclosed is my renewal. How could I say no after tha fine Jul/Aug issue? As I still consider myself a novice, that issue will be on my "active and working" pile for some time. Some top-notch authors there, including yourself. Thinking back, it was articles and code in Rainbow by Allen Huffman and Joel Hegbrg (somehow missing) that helped me get going on my first computer, a CoCo 2.

Today, I use an Amiga about 90% of the time. The other 10%, for endless learning and things like writing this letter, is devoted to a hard drive equipped 512K CoCo3 configured with RGB-DOS. You won't see many messages or letters from me, but when I do write one I use VIP Writer II. Guess I just like it!

In regard to your query about opinions concerning your support of other 68' based platforms, here is my $0.02 worth. I agree with your choice of covering only "orphan" systems. Naturally, any coverage of the Amiga would benefit me personally, but I turn to you as the ONLY printed source left for my orphan. There are many Amiga magazines and I currently subscribe to FOUR!

Please don't take this opinion as meaning you should ignore the others, however. I feel that we are all in the same family and should share important news and findings.

Larry Podowski
11 Saint Thomas St.
Pittsburgh, PA 15203

Thanks for the kudos Jim and Larry! I am going to contact someone from the Atari crowd. They are basically in the same boat as the CoCo and OS-9 family. The Mac isn't yet, though we could offer some support for the "classic" compact Macs without getting bogged down in the operating system. We'll see how it goes with Atari first. There may be no one willing to write on a regular basis, or not willing to write the types of hardware and software articles I want. We'll see.

Removeable Hard Drives Made Easy

Allen Huffman

Part 2: Setting the Syquest EZ135 up for CoCo – DECB or OS-9 Only!

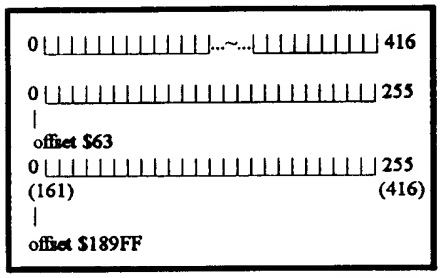
Full (DECB) Speed Ahead

If your only interest is RS-DOS, simply running RGB-DOS with it's default "OS-9 Offset" set to 0 will work fine. That will make all of your 256 RS-DOS drives start at the first sector of the hard disk, leaving 50 megs of unused space at the end. Since using an offset doesn't hurt anything, it might be a good idea to use the suggested offset of $189FF anyway. This will still give you 256 RS-DOS drives starting at a sector further into the disk, but leaves the front section available for OS-9 in the future.

Since the OS-9 offset values in RGB-DOS can actually be changed once the system is booted, you can do some interesting things with it. One EZ135 drive can hold up to 416 DECB drives, but RGB-DOS only supports 256 at a time. It would be possible to make a program to change the offset after the system is up and running, in effect "banking" you to another section of RS-DOS drives.

There is a small amount of waste we must realize before proceeding. Math lovers will notice that 630 does not divide evenly into 262,144. With 416 RS-DOS drives, there are 64 sectors left over which cannot be used. When calculating offsets, you must always based them from the start or from the end of the drive - never mixed - else regions near the center will be off by 64 sectors. If this doesn't make sense, don't worry - just read on.

Assume that each "block" in the diagram below is a 630 sector RS-DOS disk. There would be 416 of these on an EZ135 drive. By switching the offset from $189FF to $63 you could "toggle" which area of the sectors the user could access.



When the offset is $63, RS-DOS drive 0 through 255 correspond to blocks 0 through 255. When the offset is set to $189FF, drives 0 through 255 correspond to blocks 161 through 416. This would mean that no matter which offset you were using an overlap region (blocks 161 through 255) would be available to either side though drive numbers would be different. When using the first bank, copying files to drive 161 would be on drive 0 when using the second bank. This would give an easy way to transfer files back and forth between banks.

If the overlap sounds confusing (and it is), you could simply divide the drive up into two seperate sections both the same size, with no overlap. Offset $63 would be drives 0 through 207 (half of the available blocks), and an offset of $1FD6A would make drives 0 through 207 be the right half of the available blocks (blocks 208 through 415). Keep in mind that RGB-DOS doesn't "stop" it's drive numbers until it's out of room. Thus, while in the first bank, you'd actually have access to drives 208 through 255, and they would be able to access the second bank's drives 0 through 47. This would simplify the drive numbering (half and half) but still allow for overlap for transferring files back and forth. If this overlap really bothers you, memory location &H150 in RGB-DOS controls the highest accessable drive number. POKEing this to 207 would remove overlap from the equation.

To actually change this offset, POKE three memory locations inside RGB-DOS. To change the offset to $189FF, you would do as follows:

```
POKE &HD938,&H1
POKE 55608,&H1
POKE &HD939,&H89
POKE 55609,&H89
POKE &HD93A,&HFF
POKE 55610,&HFF
```

You could have a program on one of the first bank disks to do this, then another on one of the second bank disks to toggle it back, and even write some simple utilities to copy programs across the sections. I expect to have some software to do just this in the near future, but for starters here are two items to get you going.

The first program is a simple hack that selects a start or end bank as described above, complete with overlap. I put a copy of this on drive 0 of the start bank, then again on drive 0 of the end bank (which is drive 160 to the start bank). Then I RUN"BANK" when using the end bank and can switch to the start bank, or the opposite when in the start bank. (NOTE: This program isn't very graceful. It's hard coded for this particular situation.)

```
0 REM *
1 REM *  EZ135/RGB-DOS BANK Util V1.0
2 REM *       by Allen C. Huffman
3 REM * (formerly of Sub-Etha Software)
4 REM *
5 REM * This program allows brute-force
6 REM * switching between two banks of
7 REM * the 416 available drives on an
8 REM * EZ135 disk.
9 REM *
10 PRINT:PRINT"BANK [1] START OR [2]
END:";
15 A$=INKEY$:IF A$="" THEN 15
20 LN=INSTR("12",A$):IF LN=0 THEN
SOUND 200,1:GOTO 15
25 PRINT:ON LN GOTO 105,205
99 END
100 REM * Poke to START BANK
105 POKE 55608,0:POKE 55609,0:POKE
55610,63
110 PRINT:PRINT"* START BANK ACTIVE
*":END
200 REM * Poke to END BANK
205 POKE 55608,&H1:POKE
55609,&H89:POKE 55610,&HFF
210 PRINT:PRINT"* END BANK ACTIVE
*":END
```

The next program is a more graceful and flexible approach. This program requires you to edit a few variable lines so it will know about your particular hard drive. Variable "TS" in line 15 should be set to the total number of sectors your drive has. "BK" in line 20 should be set to the number of splits you wish your drive to have. Typically you want to leave this at 0 and the program will do it automatically, but if, for some strange reason, you want a 416 drive EZ disk to have 10 "splits", you can override this. "MD" in line 25 specifies if the offset should be calculated from the start of the drive or the end. For OS-9 compatibility, use offsets from the end. Finally, "OL" in line 30 controls whether or not regions can overlap. If set to 1, it will POKE the maximum drive value when the split offset is activated.

When you RUN"RGBSPLIT", the program will display drive information and the number of splits available then prompt you to select one. It will then move you to the correct offsets. One potential proble. is that since this program uses "even splits" (ie, every split has the same number of drives), rounding off can waste sec-

tors on larger drives. These wasted sectors could be turned extra mini-banks, but this program doesn't currently do this (ie, three banks of 200 with a fourth bank of 156 leftover drives). Like most things, if there is a request for this I will do my best to add it.

```
0 REM *
1 REM *    RGB-DOS SplitUtil V1.0
2 REM *      by Allen C. Huffman
3 REM * (formerly of Sub-Etha Software)
4 REM *
5 REM * This program allows switching
6 REM * between banks of RGB-DOS drives
7 REM * on hard drives with more than
8 REM * 161280 sectors ($27600).
9 REM *
10 '
11 ' * User MUST modify these values! *
12 '
13 ' TS = Total Sectors on Drive
14 '
15 TS=&H40000
16 '
17 ' BK = How Many Banks Wanted
18 '     ( 0 = automatic even split )
19 '
20 BK=0
21 '
22 ' MD = Offset Location
23 '     ( 0 = based from drive start,
24 '       1 = based from drive end )
25 MD=1
26 '
27 ' OL = Allow Drive Regions to Overlap
28 '     ( 0 = yes, 1 = no )
29 '
30 OL=0
31 '
50 REM * Calculate split(s)
55 CLS:PRINT"RGB-DOS Split
Utility.":PRINT
60 TD=TS/630 ' Total # of drives avail.
65 IF BK=0 THEN FOR BK=1 TO 999:IF
(TD/BK)>256 THEN NEXT
70 FOR SP=1 TO 999:IF (TD/SP)>256
THEN NEXT
75 DR=INT(TD/SP)
80 PRINT"Banks of drives :"BK
85 PRINT"Drives per bank :"DR
90 SW=TS-(DR*SP*630)
95 PRINT"Sectors wasted  :"SW
100 PRINT
105 PRINT"Select Bank 1 -"BK":";
110 IF BK>9 THEN 135
115 REM * Ask for single-digit bank
120 A$=INKEY$:IF A$="" THEN 120 ELSE
LN=VAL(A$):IF LN<1 OR LN>BK THEN
SOUND 200,1:GOTO 120
125 PRINT A$:GOTO 140
130 REM * Ask for multi-digit bank
135 LINEINPUTA$:LN=VAL(A$):IF LN<1 OR
LN>BK THEN SOUND 200,1:GOTO 135
140 REM * Calculate offset
145 IF MD=1 THEN OF=SW-1 ELSE OF=0
150 OF=OF+((LN-1)*630*256)
155 PRINT:PRINT"Selecting
Bank"LN"of"DR"drives."
160 L1=INT(OF/65536):A=OF-
(L1*65536):L2=INT(A/256):L3=A-L2*256
165 POKE 55608,L1:POKE 55609,L2:POKE
55610,L3
170 IF OL=1 THEN POKE &H150,DR-1
```

OS-9 Only - Hard Drive Limits

If you do not run Disk Basic, or just don't have RGB-DOS, you can easily use the EZ135 entirely for OS-9. There is a tiny problem, however. OS-9 Level 2 uses 3-bytes for accessing hard drive sectors (24-bits). This allows addressing drives with up to $FFFFFF logical sectors (4 gigabytes). However, the bitmap can only be up to $FFFF (65,535) bytes in size. Since each byte contains 8 bits, and each bit represents if a "cluster" is allocated or free, you run into a delima. OS-9 Level 2 can only use 8 * 65,535 clusters (524,280). If the cluster size is one sector, this is also the limit to how many sectors a drive can directly have without reclustering.

Reclustering involves making each bit represent more than one sector. Different cluster sizes can be used, but they must be a power of 2 (1, 2, 4, 8, etc.) With one sector per cluster, you have the least waste. A small file of only 10 bytes uses a minimum of 256 bytes. But, when you have to cluster to larger values, smaller files are more wasteful. If each allocated "cluster" were two sectors (512 bytes), the same 10 byte file would now be taking up 512 bytes on the drive.

To avoid this waste, you ideally want to use one sector clusters (the default). This limits the largest size to 524,280 sectors (65,535 bytes in the bitmap times 8 sectors per byte) or about 128 megabytes. To fully use a drive with more sectors than that you must either recluster or use a driver that supports partioning. SCSISYS 2.0 does support partioning, and can be used to make one large drive (up to the 4-gig limit) appear as several 128 meg "partitions". I will leave details of this to the SCSISYS manual. If there is enough interest, a future article may be written.

Since we have a limit of just under 128 megs, we will end up wasting about 2K bytes on each EZ disk. If the drive were significanly larger than 128 megs (such as the EZ Flyer 230 meg removeable drive), the only other options would be to recluster or partition.

To use the EZ135 entirely for OS-9, all that is required is logically formatting the unit (a low-level format is not necessary). Scsifmt is supposed to take care of this, and indeed it did work fine on my two other smaller SCSI drives. On the EZ135 it correctly reports the sector size, but does not perform a correct logical format since it wasn't programmed to deal with drives that need bitmaps larger than 64K in size. The result is that any bits over $FFFF "drop off". Thus, this drive actually needs a bitmap of $10000, which turns into a bitmap of $0000 on the drive. The end result is a bitmap of zero length, and a drive you cannot store files on. Unless scsifmt is updated to deal with drives larger than 128 megs, you will need another solution.

To get around this I simply created a descriptor for the drive with values that equal the total number of logical sectors we can use (524,280). With the aid of a calculator I came up with the following values:

sectors = 34
cylinders = 3855
(34x3855x4 = 524,280 logical sectors
sides = 4

Using dmode, the descriptor would be modified as follows:

Decimal values: dmode /h0 sct=34 tos=34 cyl=3855 sid=4

Hex values: dmode /h0 sct=$22 tos=$22 cyl=$F0F sid=$4

There is still a problem. The stock format utility apparently has trouble dealing with large drives itself. If you attempt a logical format as described earlier in this article, you will end up with a correct LSN0 sector, a bitmap the correct size, and the root directory in the right place (starting at sector $100), but all the "used" sectors for the bitmap will not be properly marked as allocated. What this means is that as you write data to the drive, it will store files on top of the area where the bitmap is located, crashing the bitmap and eventually overwriting the root directory making the disk inaccessable! I assume this problem has been reported before my discovery, and I am waiting to hear of a solution. In the meantime, we fall back on scsifmt to get the job done.

Scsifmt, as mentioned, cannot deal with large drives on it's own. If it senses the drive size, it won't work. To avoid this, when running scsifmt to do a Logical Format it will ask if it should "Query drive size (Y/N)?". Answer NO, which forces it to use the values stored in the descriptor

instead. If they have been set as listed above, scsifmt will report a size of $7FFF8 sectors and will create a properly allocated bitmap which should work fine.

After this, you will have a disk with 134,215,680 bytes of storage entirely for OS-9. This works out to $7FFF8 (524,280) useable sectors, taking into account the sectors which are used for LSN0 (the ID sector for the drive), the bitmap, and the root directory. The best part is we are only wasting 2K of the drive. If you try this using format, it will report only $FFF8 sectors, indicating that format's problem is quite possibly the same one that scsifmt has when dealing with large drives.

With the descriptor modified in this manner, all that one needs to do is a logical format as described earlier in this article. With that done, you will have 128 megs for OS-9 on each EZ disk.

RGB-DOS ROM Tips

RGB-DOS can be burnt into ROM and replace Disk Basic. It allows you to autoboot a program when the system starts up, including autobooting OS-9. To boot OS-9, the RGB-DOS drive 0 must contain a short BASIC program to perform a DOS boot from another drive. Since the low-level boot module that comes with stock OS-9 (a module called, simply enough, "BOOT") was written to load the OS9Boot file from a floppy disk in drive 0, a replacement BOOT module is needed. This module must know how to find the bootfile from the SCSI hard drive. Such a BOOT module is available, but the one that comes with the RGB OS-9 drivers is for a stock OS-9 system. NitrOS-9 users will have to seek elsewhere.

If you want OS-9 to boot completely from the hard drive using RGB-DOS, you will need it and also a utility program which correctly "links" the starting sector of the boot file on the RS-DOS partition to the OS-9 boot sector stored at the front of the hard drive. Then, when the "DOS" command is issued from an RGB-DOS partition, it loads the initial boot program from track 34 which in turn knows how to read the first sector of the drive (LSN0 for OS-9) and locate the bootfile and get OS-9 started. If interest exists, a future article will cover more details on how this works. (NOTE: A program called "LINK.BAS" is included with the RGB OS-9 drivers which does this necessary link.)

Another item worth discussing is figuring out just what offsets are best for you. If you plan to only use RS-DOS or OS-9, then your path is clear. If you plan to have disks with RS-DOS and some with OS-9, there needs to be another solution. One easy solution is to burn the rom to it will hold only one RS-DOS drived at the end of the hard disk. When using an OS-9 EZ disk, the RS-DOS image will boot OS-9. This can be done one of two ways. The first is by using the LINK program which takes an OS-9 boot disk image then splits it so an RS-DOS directory structure can also exist on the same drive. This "skitzo" disk can then contain an "AUTOEXEC.BAS" program RGB-DOS will automatically execute which will in turn perform the DOS command from that drive to boot OS-9.

The second option (and only option if you cannot make a skitzo disk as described above) is to have the AUTOEXEC program on the initial drive image simply poke the offsets to expose a second drive directly in front of it which contains an OS-9 boot disk image, then perform the DOS boot from that drive.

If you put in an EZ disk that is designed primarily for RS-DOS, this last disk image should contain a program that repokes the offsets to make the unit hold all 256 RS-DOS disks. This is not as difficult as it may sound and, once more, if there is interest a future article will be written to completely detail how to do this. The BANK and RGBSPLIT programs should offer a great starting ground for this.

My RGB-DOS is currently modified to boot up with only one RS-DOS partition, then the following AUTOEXEC.BAS program repokes the offsets to reveal a full 256 drives:

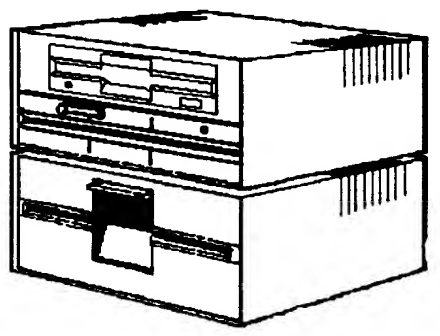
```
0 POKE65497,0 'double speed mode
1 PALETTE0,0:PALETTE8,63:PALETTE12,
63:PALETTE13,0 'set screen colors
10 REM * Identify disk
15 WIDTH40:CLS1
20 PRINT"**** RGB-DOS / OS-9 Dual EZ135
Disk ****":PRINT
25 PRINT"256 RS-DOS Disks":PRINT"50
Megabytes for OS-9":PRINT
30 REM * Setup proper offsets
35 POKE 55608,&H1:POKE 55609,&H89:
POKE 55610,&HFF:POKE &H150,255
```

One final warning for this "one drive" method. Since RGB-DOS sets the max drive value in memory location $150 on startup, if you boot without running this AUTOEXEC.BAS or by holding SPACE to bypass running it, RGB-DOS will only see one drive. If you then type DRIVE OFF to "bank out" the hard drive (turning drives 0 through 3 into your physical drives), it will still see only one drive. If you have more than one physical drive, you'd need to change the value at $150 so you can actually access physical drives other than 0. One solution might just to have the ROM set to four drives. Perhaps a bit more wasteful if you don't plan on using four drives for RS-DOS on your OS-9 EZ disk, so I'll let you decide which approach is suitable.

Conclusion

With the price of the EZ135 recently lowered to about $130, there simply is no nicer hard drive solution for a CoCo owner. The drive should be compatible with all SCSI adapters for the CoCo, including Ken-Ton, Disto and LR-Tech and can also be used on any other computer with SCSI (PC, Mac, Atari ST, Amiga, etc.). The media size is perfect, and the speed is faster than what a CoCo could support anyway (and even faster than what larger OS-9 systems such as the MM/1 can handle). Speed tests show the EZ135 and SCSISYS 2.0 able to transfer one megabyte is less than 13 seconds, which is about as fast as it gets with current software. Also, the wide availability of both the EZ135 drive and extra disks makes it an appealing "off the shelf" solution for unlimited storage.

Major stores such as Best Buy and CompUSA should have the unit in stock and, in fact, CompUSA recently had a $30 rebate bringing the end price down to $99. A bargain by any means. For more details as well as a comparision between the SyQuest EZ135 and Iomega ZIP drives, please see the last issue.



FARNA Systems

Your most complete source for Color Computer and OS-9 information!

Post Office Box 321
Warner Robins, GA 31099
Phone: 912-328-7859
E-mail: dsrtfox@delphi.com

ADD $3 S&H, $4 CANADA, $10 OVERSEAS

BOOKS:

Mastering OS-9 - $30.00
Update of Paul Ward's "Start OS-9". Completely steps one through learning all aspects of OS-9 on the Color Computer. Easy to follow instructions and tutorials. Comes with a disk full of added utilities and software!

Tandy's Little Wonder - $25.00
History, technical information, hacks, schematics, repairs,... almost EVERYTHING available for the Color Computer all under one cover! A MUST HAVE for those wanting to keep their CoCo alive or for those new to the CoCo.

Quick Reference Guides
These handy little books contain the most referenced information in an easy to find format. Small size makes them unobtrusive on your desk. Command syntax, error codes, system calls, etc.
CoCo OS-9 Level II - $5.00
OS-9/68000 - $7.00

SOFTWARE:

CoCo Family Recorder: The very best in genealogy record keeping EVER for the CoCo! Requires CoCo3, two disk drives (40 track for OS-9) and an 80 column monitor.
DECB: $15.00 OS-9: $20.00

DigiTech Pro: $10.00
Easily add sounds to your BASIC and M/L programs! Very easy to use. Requires user to make a simple cable for sound input through a joystick port. Requires CoCo3, DECB, 512K.

ADOS: The premiere, most respected enhancement for DECB! Support for double sided drives, 40/80 tracks, faster formatting, many extra and enhanced commands!
Original (CoCo 1/2/3) - $10.00
ADOS 3 (CoCo 3 only) - $20.00
Extended ADOS 3 (CoCo 3 only, requires ADOS 3, support for 512K-2MB, RAM drives, 40/80 track drives mixed) - $30.00
ADOS 3/EADOS 3 Combo: $40.00

Pixel Blaster - $12.00
High speed graphics tools for CoCo 3 OS-9 Level II. Easily speed up performance of your graphics programs! Designed especially for game programmmers!

Patch OS-9 - $7.00
The best enhancement for OS-9 short of buying a 6309 and Nitr0S9! Contains the latest version of all popular utilities and new commands with complete documentation. Has auto-installer (requires 2 40T DS drives, one may be larger). Ask for 35 track disks (manual install).

HARDWARE:

SPECIAL PRICE ON REMAINING VIDEO DIGITIZERS: $100 (was $125)
This may be your LAST CHANCE to get a DigiScan video digitizer! No MPI -- uses joystick ports. CoCo Max3 , Max 10, Color Max 3 compatible. Input from any VCR, camcorder, or TV camera (freeze-frame required).

Special Prices on Remaining DISTO stock!
LIMITED QUANTITIES! BUY NOW!
Mini Controllers: Clones of the Tandy short disk controller, comes with DECB 2.1. NO CASE. $40 each
512K Upgrades: With RAM chips: $50, without: $25
Super Controller I : I have several boards that need repair. Buy two for $30 so you'll have extra parts, or one for $20. Comes with schematic. NO WARRANTY.
HD Controller: $30 - Connect embedded SCSI or MFM/RLL drives (with Adaptec controller) to your CoCo. These fit inside a Super Controller I or II.
Bare Board Set: $30 - Set of blank circuit boards for Disto products including SCI, SCII, HD, 4-n-1, MPROM, etc. Add $20 for complete schematic set.
Complete Disto Schematic set: $25.

FARNA Systems AT306 Based Computers

Complete computer systems based on the AT306 board from Kreider Electronics. Systems are completely setup and ready to go, just add a monitor (or we can supply that too)!

Both systems include:

16 bit PC/AT I/O bus with five slots
MC68306 CPU at 16.67MHz
4 30 pin SIMM sockets
IDE Hard Drive Interface
1.4MB Floppy Drive
Two 16 byte fast serial ports (up to 115K baud)
Bi-directional parallel printer port
Real-time clock
PC/AT keyboard
Desktop Case and Power Supply
BASIC (resembles Microsoft BASIC)
MGR Graphical Windowing Environment with full documentation
"Personal" OS-9/68000 Vr 3.0 (Industrial with RBF)
Drivers for Tseng W32i and Trident 8900 VGA cards
Drivers for Future Domain 1680 and Adaptec AAH15xx SCSI cards

Many other utilities and tools

FARNA-11122 Includes:
2MB RAM
200MB Hard Drive*
Trident 8900 1MB Video Card
$960.75

FARNA-11225 Includes:
2MB RAM
500MB Hard Drive*
Tseng W32i 1MB Video Card
$1114.47

****This is the SMALLEST amount of formatted space (most manufacturers specify UNFORMATTED CAPACITY). Prices fluctuate - we get you the largest drive possible for the money allotted!***

Call for a quote on different configurations and components. Warranty is 90 days for labor and setup, components are limited to manufacturers warranty.

Microware Programmers Package -
Licensed copies of Microware C compiler, Assembler, Debugger, Basic, and many other tools!
***With system purchase:* $65.00 *Without system:* $85.00**

More software will be listed later!

Bomb Sweeper

A "minefield" game for the CoCo3

Joseph A. Pellettiere

The program is quite simple and works just like the popular, time consuming game on a Windows® system. It requires a CoCo3 with a two button joystick or mouse, although I'm sure the second button could easily be mapped to a key.

You have an 8x8 grid. Under the squares of the grid are a certain number of bombs (the current number is shown in the left hand side of the screen). You push the left button to uncover a square. Three things can happen when you do this: either the square is blank, there will be a number, or there will be a bomb. If there is a bomb, the game is over. If there is a number, then this number represents the number of bombs occupying the surronding squares touching the numbered square. If the square is blank then there are no bombs touching the square.

Using the second button marks the square as containing a bomb, and decreases the bomb count by one. If you click on a marked square it goes back to its default. You can mark and unmark unturned squares, but once a square is turned over, that's it.

All the while you are doing this a timer in the right corner is keeping track of your time. So the goal would be try to uncover all the squares as quickly as you can. You can press break, the start or quit.

The program is mostly done, and in my opinion runs pretty good. The code is a little spaghetti like, but I wrote it in parts over several months and each time I had to remember what I did before. I never really set up the colors, sounds, ending, or some shortcuts. Maybe if there is significant interest I will continue and have another update ready. Send me any comments/hints/thing you'd like to see in a future release to me via e-mail at jap5r@virginia.edu, or send a letter to the editor of this magazine or the address in the first few lines of code.

```
1 REM THIS IS POSTCARD WARE, IF YOU
LIKE THIS PROGRAM SEND A
2 REM POSTCARD TO BRITTANY
PELLETTIERE 302 4SEASONS DR
3 REM CHARLOTTESVILLE VA 22901 OR
COMMENTS TO
4 REM JAP5R@VIRGINIA.EDU
6 HBUFF1,380:HBUFF2,380:HBUFF3,380:
HBUFF4,380:HBUFF5,28:HBUFF6,28
10 POKE65497,0:GOSUB5000
15 ON BRK GOTO 1150
20 XX=RND(-TIMER)
30 DIM GR(10,10):DIM GS(10,10)
35 W$=""
40 FOR I=1TO10:FORJ=1TO10:GR(I,J)=0:
GS(I,J)=1
50 NEXTJ:NEXTI
55 NM=0:GU=0
60 X=RND(7)+2:Y=RND(7)+2:IFGR(Y, X)<0
GOTO 60
70 GR(Y,X)=-1:NM=NM+1:IF NM<9 GOTO 60
100 FORI=2TO9:FORJ=2TO9
105 CN=0
110 IF GR(I,J)=-1 THEN 190
120 FOR II=I-1TOI+1:FORJJ=J-1TOJ+1
130 IF II=I AND JJ=J THEN 140
135 IF GR(II,JJ)=-1 THEN CN=CN+1
140 NEXTJJ:NEXTII
150 GR(I,J)=CN
190 NEXTJ:NEXTI
500 REM
501 NM=9
505 HSCREEN2:POKE&HFF9A,16: HCLS8
506 IF W$="Y" THEN 700
509 HCOLOR4,8
510 HLINE(79,40)-(93,40),PSET,B
511 HLINE(79,40)-(79,50),PSET,B
515 HCOLOR5,7
520 HLINE(80,41)-(93,50),PSET,BF
530 HGET(78,39)-(94,51),1
571 HCOLOR 8,8
575 HLINE(82,48)-(94,49),PSET,BF
577 HDRAW"BM85,48C3U7F3G3"
578 HPAINT(86,45),3,3
580 HGET(78,39)-(94,51),2
600 HCOLOR4,8
610 HLINE(79,40)-(93,50),PSET,BF
620 HGET(78,39)-(94,51),3
630 HCIRCLE(87,45),5,8
640 HPAINT(85,45),8,8
650 HGET(78,39)-(94,51),4
660 HCOLOR8,8
670 HLINE(78,39)-(96,51),PSET,BF
700 FORI=0TO7:FORJ=0TO7
710 II=100+16*I:JJ=48+J*12
720 HPUT(II,JJ)-(II+16,JJ+12),1
730 NEXTJ:NEXTI
740 HCOLOR3,8
750 HLINE(99,47)-(230,145),PSET,B
1000 HGET(16,16)-(20,20),6
1010 HCOLOR3,8
1020 HLINE(16,16)-(20,20),PSET,BF
1030 HGET(16,16)-(20,20),5
1040 HPAINT(17,17),8,8
1045 T=TIMER:TI=0:OT=0
1047 HPRINT(12,3),"BOMBS      TIME"
1050 X=16:Y=16:X1=16:Y1=16
1055 HPRINT(13,4),9
1056 HPRINT(15,22),"BOMBSWEEPER!!":
HPRINT(11,23),"BY JOSEPH PELLETTIERE"
1060 'BEGIN GETTING POSITIONS
1062 IF GU=64 AND NM=0 THEN 4500
1064 TI=INT((TIMER-TT)/60)
1065 IF TI=OT THEN 1070 ELSEOT=TI
1066 HCOLOR3,8:HPRINT(25,4),TI-1:
HCOLOR3,8:HPRINT(25,4),TI
1070 IF X1=X AND Y1=Y THEN 1120
1080 HPUT(X1,Y1)-(X1+4,Y1+4),6
1090 HGET(X,Y)-(X+4,Y+4),6
1100 HPUT(X,Y)-(X+4,Y+4),5,PSET
1110 X1=X:Y1=Y
1120 IF BUTTON(0)=1 THEN1200ELSE IF
BUTTON (1)=1THEN1700ELSE1990
1150 HPRINT(2,10),"ANOTHER"
1160 HPRINT(2,11),"GAME (Y/N)"
1170 W$=INKEY$:IFW$="Y"THEN 40
1180 IF W$="N" THEN 6000
1190 GOTO1170
1200 IFX<99ORX>228ORY<47ORY>144
THEN1060
1210 GOSUB 2020
1215 IF GS(XI+1,YI+1)<>1 THEN 1990
1216 GU=GU+1
1219 PLAY"O1L8E-"
1220 MK=GR(XI+1,YI+1)+2
1230 IF MK>1 THEN 1550
1500 HPUT(X,Y)-(X+4,Y+4),6:HPUT(XP,YP)-
(XP+16,YP+12),4
1505 GOSUB 2010
1540 GOTO 4000
1550 HPUT(X,Y)-(X+4,Y+4),6:HPUT(XP,YP)-
(XP+16,YP+12),3
1553 N=MK-2
1555 IF N<1 THEN 1570
1560 XL=INT((XP+6)/8)-1:YL=INT((YP+ 6)/8):
HPRINT(XL,YL),N
1570 GOSUB 2010
1580 GS(XI+1,YI+1)=3
1590 GOTO1990
1700 IFX<99ORX>228ORY<47ORY>144
THEN1060
1710 GOSUB 2020
1715 S=GS(XI+1,YI+1):ON S GOSUB
1730,1740
1720 GOTO1060
1730 HPUT(XP,YP)-(XP+16,YP+12),2:GS(XI
+1,YI+1)=2:GOSUB2010:HCOLOR
8,8:HPRINT(13,4),NM:HCOLOR3,8:NM=NM-
1:HPRINT(13,4),NM:GU=GU+1:RETURN
1740 HPUT(XP,YP)-(XP+16,YP+12),1:GS(XI
+1,YI+1)=1:GOSUB2010:HCOLOR8,8:HPRINT
(13,4),NM:HCOLOR3,8:NM=NM+1:HPRINT(13,
4),NM:GU=GU-1:RETURN
1990 X=JOYSTK(0)*5:Y=JOYSTK(1)*3
2000 GOTO1060
2010 HGET(X,Y)-(X+4,Y+4),6:HPUT(X,Y)-(X+
4,Y+4),5:RETURN
2020 XI=INT((X-100)/16)+1:YI=INT((Y-48)/
12)+1
2030 XP=100+16*(XI-1):YP=48+12*(YI-1):
RETURN
4000 PLAY"A;A#;A-"
4010 HPRINT(30,10),"KABOOM"
4020 HPRINT(30,11),"YOU LANDED"
4030 HPRINT(30,12),"ON A BOMB"
4090 GOTO4090
4500 PLAY"AAAAAAAA"
4620 HPRINT(30,10),"Congrats"
4630 HPRINT(30,11),"You found"
4640 HPRINT(30,12),"The bombs!
4700 GOTO 4700
5000 CLS0
5010 POKE &HFF02,0:Z=PEEK(&HFF00)
5020 IF Z<>127ANDZ<>255THEN PRINT"
RELEASE KEYS":GOTO5010
5050 POKE &HADEB,57
5060 READ A:IFA=999THEN5080
5070 POKE
&HDF60+X,A:X=X+1:GOTO5060
5080 POKE &HA1C2,223:POKE &HA1C3,96
5090 POKE &HDF67,Z:POKE &HADEB, 189
5140 RETURN
5150 DATA 127,255,2,182,255,0,129, 255,
38,1,57,126,161,203,999
6000 POKE65496,0:END
```

68HC11 Microcontroller Programming

Geoff Cusick

Why use anything other than C for embedded applications?

Editor: *This short article was written in response to a request I made for a comparison of the 6809 and 68HC11 programming models. I was mainly interested in assembler level code, as many readers are familiar with 6809 assembly language.*

If anyone has any other thoughts on this subject, please let me know. I would also be interested in any other programming techniques and tips concerning the 68HC11 or other Motorola microcontrollers so this series could continue as a regular feature. Hardware and programming projects are also welcome.

I used to write a lot of assembler for the 6809 some years back now, and there is no doubt that it is a very easy processor to program in assembler. In particular, writing "structured" and re-entrant code is dead easy, with the comprehensive stack-handling facilities.

But then the 'C' compiler came along. I made the transition still programming 6809's, all in custom hardware, and spent a fair amount of time writing assembler-to-C hooks, and vice versa, since the first compiler we had (from Hewlett Packard) produced bizarre machine code, which really couldn't be used in real-time programs. Then I got a better 'C' compiler (Whitesmiths), and progressively, the need to write assembler diminished.

About 2 years back, we decided that we needed to move on to single-chip microcontrollers, and, rightly or wrongly settled on the 68HC11. From the word 'go', there was no intention to program in anything other than 'C', and I confess that the number of lines of assembler I've written since could be counted on the fingers of one thumb. Early on, I looked at the assembler output from the compiler (IAR), just to convince myself that it wasn't producing code in the Hewlett Packard model.

Now the only assembler programming I do is to hack the startup code to match particular hardware requirements - it's really pretty mindless, and for the most part, it's possible to write programs without any real knowledge of the CPU or its instruction set. The compiler produces good, compact code, which is perfectly adequate for the sort of real-time measurement and control applications we've got.

The development system we use is PC hosted. It comprises the IAR compiler-linker suite (which includes an assembler), and an Ashling In-Circuit Emulator. Total cost was about 5000 sterling.

At just about every _visible_ level the 6809 and 'HC11 are very different. Most of our applications use the 'HC11 in "single-chip" mode, with no external memory. This means that we can embed a processor (cheaply) in small, portable instruments. The 'HC11 has features like an on-board async comms port, SPI, a-d converter and parallel ports which would all need additional packages with the 6809, and which are a major advantage in the instrumentation applications we have. You just have to adapt the programming style a little to work in 512 bytes of RAM!

'C' compilers, particularly for the modern mainstream processors are cheap, and reasonably efficient. Outside the hobby market, I'm not sure that assembler language programming has much future - it's lengthy to write, prone to error and difficult to debug in a structured way.

Geoff Cusick
Department of Medical Physics &
Bioengineering, UCL Hospitals,
The Middlesex Hospital London
W1N 8AA

Tel: (+44) 171 636 8333 ext 4266
Fax: (+44) 171 380 9111
E-mail: geoff@medphys.ucl.ac.uk

ATTENTION! ATTENTION! ATTENTION! ATTENTION! ATTENTION!

You are cordially invited to the first Pennsylvania

COLOR COMPUTER SHOW & SALE

WHEN? August 2 & 3, 1997 (Sat. 10am-5pm; Sun. 10am-3:30pm)
WHERE? EMBERS INN 1700 Harrisburg Pike, Carlisle, PA

ADMISSION:
$5.00 per person per day *OR* $7.00 for both days (paid in advance). Children accompanied by a responsible adult are FREE!

HOW TO GET THERE:
Exit 16 off of the PA Turnpike I-76, turn to Harrisburg, go 1.3 miles, it's on the right *OR* I-81 Exit 17, turn left, go 1/10th mile, on the right!)

Overnight room rate:
$60 - Be sure to ask for the "FEST" rate! Call 1-717-243-1717 OR 1-800-692-7315 OR Fax (717) 243-6648 for reservations! There is a limited supply of rooms blocked out for the show. Reserve your room early - - these rooms will be released for regular reservations on July 1, 1997 and will NOT be available at the show rate after that date!

For further information, General or Exhibitor, contact:

Ron Bull
115 Ann Street
Duncannon, PA 17020

Phone: 717-834-4314
E-mail: ronbull@aol.com
OR
ron.bull@paonline.com

HawkSoft

28456 S.R. 2, New Carlisle, IN 46552
219-654-7080 eves & ends MO, Check, COD; US Funds
Shipping included for US, Canada, & Mexico

MM/1 Products (OS-9/68000)

CDF $50.00 - CD-ROM File Manager! Unlock a wealth of files on CD with the MM/1! Read most text and some graphics from MS-DOS type CDs.

VCDP $50.00 - New Virtual CD Player allows you to play audio CDs on your MM/1! Graphical interface emulates a physical CD player. Requires SCSI interface and NEC CD-ROM drive.

KLOCK $20.00 - Optional Cuckoo on the hour and half hour!! Continuosly displays the digital time and date on the /term screen or on all open screens. Requires I/O board, I/O cable, audio cable, and speakers.

WAVES vr 1.5 $30.00 - Now supports 8SVX and WAV files. Allows you to save and play all or any part of a sound file. Merge files or split into pieces. Record, edit, and save files; change playback/record speed. Convert mono to stereo and vice-versa! Record and play requires I/O board, cable, and audio equipment.

MM/1 SOUND CABLE $10.00 - Connects MM/1 sound port to stereo equipment for recording and palyback.

GNOP $5.00 - Award winning version of PONG(tm) exclusively for the MM/1. You'll fo crazytrying to beat the clock and keep that @#$%& ball in line! Professional pongists everywhere swear by (at) it! Requires MM/1, mouse, and lots of patience.

CoCo Products (DECB)

HOME CONTROL $20.00 - Put your old TRS-80 Color Computer Plug n' Power controller back on the job with your CoCo3! Control up to 256 modules, 99 events! Compatible with X-10 modules.

HI & LO RES JOYSTICK ADAPTER $27.00 - Tandy Hi-Res adapter or no adapter at the flick of a switch! No more plug and unplugging of the joystick!

KEYBOARD CABLE $25.00 - Five foot extender cable for CoCo 2 and 3. Custom lengths available.

MYDOS $15.00 - Customizable, EPROMable DECB enhancement. The commands and options Tandy left out! Supports double sided and 40 track drives, 6ms disk access, set CMP or RGB palettes on power-up, come up in any screen size, Speech and Sound Cartridge support, point and click mouse directory, and MORE OPTIONS than you can shake a stick at! Requires CoCo3 and DECB 2.1.

DOMINATION $18.00 - Multi-Player strategy game. Battle other players armies to take control of the planet. Play on a hi-res map. Become a Planet-Lord today! Requires CoCo3, disk drive, and joystick or mouse.

SMALL GRAFX ETC.

"Y" and "TRI" cables. Special 40 pin male/female end connectors, priced EACH CONNECTOR -	$6.50
Rainbow 40 wire ribbon cable, per foot -	$1.00
Hitachi 63C09E CPU and socket -	$13.00
512K Upgrades, with RAM chips -	$72.00
MPI Upgrades	
For all large MPIs (PAL chip) -	$10.00
For all small MPIs (satellite board) -	$10.00
Serial to Parallel Convertor with 64K buffer, cables, and external power supply -	$50.00
2400 baud Hayes compatible external modems -	$40.00

ADD $2.00 S&H TO EACH ORDER

SERVICE, PARTS, & HARD TO FIND SOFTWARE WITH COMPLETE DOCUMENTATION AVAILABLE. INKS & REFILL KITS FOR CGP-220, CANON, & HP INK JET PRINTERS, RIBBONS & vr. 6 EPROM FOR CGP-220 PRINTER (BOLD MODE), CUSTOM COLOR PRINTING.

Terry Laraway
41 N.W. Doncee Drive
Bremerton, WA 98310
206-692-5374

The BlackHawk MM/1b

- Based on the AT306 board from Kreider Electronics. Features buil into the motherboard include:
- 16 bit PC/AT I/O bus with five slots
- MC68306 CPU at 16.67MHz
- 512K to 16MB of RAM with 30 pin SIMMs (4 sockets)
- IDE Hard Drive Interface (2 drives)
- 360K-1.44MB Floppy Drive Interface (2 drives)
- Two 16 byte fast serial ports (up to 115K baud)
- Bi-directional parallel printer port
- Real-time clock
- PC/AT keyboard interface
- Standard PC/AT power connector
- Baby AT size - fits standard PC case
- BASIC (resembles Microsoft BASIC)
- MGR Graphical Windowing Environment with full documentation
- "Personal" OS-9/68000 Vr 3.0 (Industrial with RBF)
- Drivers for Tseng W32i and Trident 8900 VGA cards
- Drivers for Future Domain 1680 and Adaptec AAH15xx SCSI cards
- OS-9/68000 Vr 2.4 with Microware C 3.2, Assembler, MW Basic (like Basic09), MW Debug, MW Programmers Toolkit
- UUCP from Bon Billson
- Ghostscript (software PostScript interpreter)
- Many other utilities and tools

Prices start at $400!
(motherboard,
Personal OSK, & MGR only)

BlackHawk
Enterprises, Inc.
756 Gause Blvd. #29
Slidell, LA 70458
E-mail: nimitz@delphi.com

Operating System Nine

Alan DeKok

Some ins and outs of OS-9 from the NitrOS-9 programmers!

This article is one of a series on the inner workings of OS-9 Level II and NitrOS-9. We will start with some general OS-9 procedures, but may mention some NitrOS-9 specifics as we go along. The articles are written by Alan DeKok and Colin McKay, who developed the NitrOS-9 "upgrade" that speeds up OS-9 Level II on the Tandy Color Computer. Many of these procedures are similar on OS-9/68K computers.

The OS-9 Boot Process

Ever wonder what happens between the time you type DOS under Disk Basic, and OS-9 makes its magical appearance? In this article we detail the steps taken when booting OS-9, and in later articles we will cover debugging OS-9 boot disks and creating OS-9 boot disks.

This article began as a result of a discussion on Delphi. Many people expressed an interest in knowing exactly what is going on inside their Coco, so after a short discussion, this series of articles was written. In this series, we make many references to the OS-9 manuals, where "Tech Ref" and "Commands Ref" are the Technical and Commands Reference sections of the OS-9 Level II manuals respectively.

1. Booting OS-9 usually starts when you type DOS <ENTER> under Disk Basic. On a floppy system, the drive heads move to Track 34 Side 0, and 18 sectors are read into Disk Basic memory starting at $2600. In Disk Basic, this is in block $39, extended address $072600 (Tech Ref 2-10). Hard drives have the kernel track located elsewhere, but it is still loaded into address $2600.

This track contains REL, BOOT, and OS9P1. In addition, just before REL is a six byte header. The structure of the header under OS-9 is shown in Table 1, and breaks down as follows.

The characters "OS" (Operating System), which act as a flag to the Disk Basic DOS command. This flag indicates that the following information on the boot track is valid.

The next two bytes are hex $20 $xx, where the '$xx' depends on which version of REL you are running. These bytes are the machine language instructions 'BRA PC+xx'. When the DOS command loads in track 34 and the "OS" string exists at the start, it transfers control of the computer to the program instructions that immediately follow the "OS". In this case, the code BRAnches to the entry point of REL, skipping over the OS-9 module header for REL.

The "OS" string and the code that gets executed after it are also the key to the Disk Basic program from Rainbow Magazine a few years ago (A Special Use for the DOS Command, Roger Schrag, Nov 1984, p140) that lets you run DECB programs by typing DOS. A special machine language program is stored on track 34, which executes Disk Basic commands the user previously specified.

$0000-$0001	"OS"	flag to DOS command
$0002-$0003	$20 $xx	BRA to execution start of REL
$0004-$0005	$12 $05	filler (flag?) bytes
$0006-$012F	REL	initialization and setup module
$0130-$02FF	BOOT	module which loads OS9Boot
$0300-$11D9	OS9p1	* half of the OS-9 kernel
$11DA-$11EC	$39 $39...*	filler bytes.
$11ED	$55	flag to ROMs that warm reset is allowed
$11EE-$11F0	SWI3	vector (LBRA to inside of OS9p1)
$11F1-$11F3	SWI2	vector (LBRA to inside of OS9p1)
$11F4-$11F6	FIRQ	vector (LBRA to inside of OS9p1)
$11F7-$11F9	IRQ	vector (LBRA to inside of OS9p1)
$11FA-$11FC	SWI	vector (LBRA to inside of OS9p1)
$11FD-$11FF	NMI	vector (LBRA to inside of OS9p1)

* NitrOS-9 has a slightly longer OS9P1, and uses different filler bytes. All other table entries apply to both OS-9 Level II and NitrOS-9..

Table 1 — Structure of the kernel track under Stock OS-9

The next two bytes, just before the start of the REL module are hex $12 $05. These do not appear to have any purpose on the Coco.

2. When control is transfered to the REL module, REL shuts off all interrupts, maps block $00 into memory at address $0000, and clears out the system direct page. It then sets up and clears the boot screen, prints the startup message, and in the case of NitrOS-9, checks for a 6309 and puts it into Native (6309) mode.

REL then checks for a cold or warm boot. On a cold boot, REL needs to copy the boot track to high memory at $ED00, block $3F, and jumps to a routine inside the new copy of REL. On a warm boot, the boot track has already been copied to high memory. Once at high memory, REL sets up the OS-9 reset (D.CBStrt) and crash (D.Crash) handling code that allows it to re-boot or to print the 'FAILED' message. REL then jumps to OS9P1. REL contains the 'OS-9 BOOT' or 'Welcome to NitrOS-9' message, as well as the infamous 'FAILED' message.

Note that by copying the boot track to high memory, the SWI, SWI2, NMI, etc. (Tech Ref 2-16) vectors in Table 1 automatically override the vectors that were there previously from DECB.

3. OS9P1 sets up the system direct page variables, and clears out the rest of low memory. See the DEFS/os9defs file on the OS-9 Development System disk, side B for information on the D.xxx variables. OS9P1 then sets up the default system DAT image in memory and the MMU, and sets up the system calls supported by OS9P1. Next, OS9P1 checks the memory size of the system, and then does a F$VModul (Tech Ref, page 8-111) on REL, BOOT, and OS9P1. OS9P1 then links to and calls the BOOT module.

4. BOOT reads LSN0 (Logical Sector Number 0), usually from drive 0. Some versions of BOOT provided with specific hard drive systems, will read the boot information from a hard drive. If LSN0 does not contain a pointer to the location and size of the OS9Boot file, BOOT crashes the system, and you get the helpful 'FAILED' message.

Boot then makes a system call to ask OS9p1 for enough memory for the OS9Boot file, and loads the OS9Boot file into that memory. BOOT saves pointers in the system's direct page to the OS9Boot file D.BtSz and D.BtPtr, and returns to OS9p1. These pointers are used by OS9p1 to find the OS9Boot file, so it can verify

the modules and add them to the module directory. These pointers are also used by the 'cobbler' program, which writes a copy of the currently OS9Boot file from memory out to disk.

5. OS9P1 then verifies the modules in OS9Boot.

6. OS9P1 checks for the presence of INIT. INIT, for all practical intents and purposes, may be thought of as a device descriptor for the computer itself. It is really just a table of preferred settings for the machine, such as the default device (/DD) to be used. (Tech Ref A-3)

7. OS9p1 next calls OS9P2, which adds its system calls to the system call table. (Tech Ref 8-107, F$SSvc)

8. OS9p2 does a CHX to the default device pointed to by the INIT module (usually /DD or /D0).

9. The default output device (/Term) pointed to by INIT is then opened. If things fail here (i.e. 'FAILED' message), and you are running a GrfInt/WindInt screen, it is probably because GrfDrv either isn't present in /DD/CMDS, or does not have its execution attribute set. If you are writing to a VDG type screen, you won't get a FAILED message because GrfDrv isn't used.

10. OS9P2 tries to link to OS9p3, and calls OS9p3 if it exists. OS9p3 is then supposed to call OS9p4, etc. These calls provides for extensions to OS-9 such as the OS9p3 that prints out complete error messages rather than just the error numbers. (Tech Ref 2-1)

11. CC3Go is then entered, and does:

CHX /CMDS
(ie. /DD/CMDS if INIT points to /DD)
CHD /DD
(this is why you MUST have a DD descriptor)
CHX /DD/CMDS

The CHX is done twice in case one fails so the boot will continue. (For example if INIT contained /D0 rather than /DD.)

Note that CC3Go is a program (not a system module), and thus does not need to be a part of the OS9Boot file. Instead you can put it in the root of the boot device (/d0/cc3go) with execution attributes set and save some system ram.

12. CC3Go then makes assorted system calls to find out internal information about the system (ie. memory etc.), and does an F$STime which results in OS9p2 calling Clock and initializing multitasking.

At this point, hitting reset will restart the whole process at step 2, so this could be considered the point where the bootstrap process ends.

13. CC3Go next tries 'SHELL Startup'. An error here is non-critical. If there is no startup file, or shell does not exist, an error is returned, or the first module in the shell file is executed, and things move on to step 14. The first module in the '/dd/cmds/shell' file is considered to be your shell (ie. command line processor).

14. Finally, the shell prompt appears, and things are ready to go.

If a real shell isn't present, then the firs module in the shell file is executed. Once that program exits, things will appear to be locked, as OS-9 has no new tasks to switch to.

As an experiment, try the following on a BACKUP boot disk. Rename SHELL to SHELL.REAL, and rename TREE to SHELL. When CC3Go loads in SHELL (TREE), it will then consider TREE to be the shell it will use. Make sure you have a backup boot disk before you try this!

In the case of TREE as your shell in the last step, 'TREE Startup' will be executed, and any output sent to the screen. Once TREE ends, control returns to OS-9. As mentioned in step 14, because there are then no tasks to switch to, the system will appear to be locked.

Hopefully this clears up any questions you may have had about the boot process. If you have any further questions, just write c/o 268'm or via internet email to 'aland@sandelman.ocunix.on.ca', and I'll try to answer them.

MM/1 Hard Disk Partitioning

Eddie Kuns

OS-9/68000 does not directly support partitioning of a hard drive. It can, however, be accomplished by modifying the drivers. Below are my partitions and the makefile I used to build them. I modified h0a.a, h0b.a, and h0c.a which came with my MM/1 to build the partitions. These same techniques should work with other systems as well, provided source code is available.

Notice the following fields in the descriptor source files:

CtrlrID - SCSI device number of this disk, same for all partitions!

PartNo - The partition number. MUST BE between 1 and 15 (inclusive).

Cylnders - The number of cylinders for this partition.

Heads - "arbitrary" number

SectTrk & SectTrk0 - "arbitrary" number - must be the same number

LSNOffset - Sector offset on disk to first sector of this partition

TotalCyls - Total # cyls of all partitions on the disk

DrvNum - Seems to always be zero. I don't understand this field.

SectSize - I use 512 bytes/sector. You probably want to also.

To partition your disk, first decide how many partitions you want and roughly how large you wish each to be. You know the total number of sectors on the disk (I assume), so choose arbitrary numbers for the number of heads and sectors/track to give an integral number of cylinders on the disk. My hard disk has 205560 512-byte sectors. I chose to divide this into 571 cylinders. 205560/571 = 360 sectors per cylinder, so I chose 8 heads and 45 sectors/track.

How you choose to divide this up probably does not matter at all. I believe the driver always uses only LSN, and never head or cylinder. I felt safer splitting it so that the number of heads was a small number. Thus, I can allocate partitions in 180k increments. (45 * 8 * 512 bytes) I chose 350 cylinders for partition 1 (/dd, /h0, 63 meg), 150 cylinders for partition 2 (/spool, 27 meg), and the remaining 71 cylinders for partition 3 (/usr, 12.78 meg).

For partition 1, Cylnders = 350, LSNOffset = 0, TotalCyls = 571

For partition 2, Cylnders = 150, LSNOffset = 126000, TotalCyls = 571

For partition 3, Cylnders = 71, LSNOffset = 180000, TotalCyls = 571

126000 = 45 * 8 * 350
180000 = 45 * 8 * (350 + 150)

Calculate LSNOffset carefully; I don't believe anything ever checks to make sure partitions do not overlap! (I could be wrong. I never tried to deliberately overlap partitions.) If someone is sufficiently paranoid, one could write a program to write a specific pattern in the last sector of each partition, and then check all other partitions for that pattern. (To be run right after formatting and before writing files to the disk!)

The only different fields for the different partitions of a disk are: "nam", PartNo, Cylnders (if you choose), and LSNOffset. All the rest should be the same. Notice that the makefile is where I specify the names of the disks. I make multiple versions of some partitions with different names. /DD and /H0 are the same disk, for example.

To set up the partitions, first do a physical format on hard disk as one device. Then boot with (or just load) the new partitions. Now do a LOGICAL format on each partition. From now on just boot with your new bootdisk with the partitioned *instead* of the non-partitioned descriptor, and you're set!

Internet: ekuns@kilroy.chi.il.us
Delphi: eddiekuns

Working with Multi-Vue

Install Tandy's GUI on a hard drive for best speed

Chris Burke & Jim Johnson

Editor: *Sometimes it seems like Tandy tried to really do right by the CoCo. They created the CoCo 3, updated OS-9 to Level 2, then announced that only OS-9 programs would be considered by them for sale. They even marketed some third party hardware and software for a while.*

One of the shining examples of software to come from Tandy was Multi-Vue. This terrific (for the time) graphical user interface (GUI) was designed to make OS-9 a lot easier to use for the average person. There were, however, a couple problems with it.

The first was the manual. Not a whole lot of detail there. It was obviously written by someone already familiar with OS-9... a glaring mistake that most OS-9 products share. The second was that Tandy didn't support a decent hard drive system for the CoCo. Luckily, several third party hard drives were available... the Burke & Burke CoCoXT was popular, and so were several SCSI controllers.

Why was a hard drive so important? OS-9 really needs lots of fast storage space, especially when complicated programs such as Multi-Vue are being used. Multi-Vue really shines on a hard drive.

Unfortunately, no hard disk controllers are made for the CoCo any more. A couple people are working on this. In the meantime, there are a good many on the used market. You'll have to find them where you can, just keep your ears open and let people know you are looking for one.

Another problem is that Tandy never told anyone how to put Multi-Vue on a hard drive. Given the awful documentation that came with Multi-Vue to begin with, it probably wouldn't have been very good.

This article will solve the last problem. Hopefully, it will help those who currently have hard drives. Those of you still looking need to keep this handy!

Installing Multi-Vue on a Hard Disk

Well, I just finished installing Multi-Vue on my CoCo 3 development system. The manual does not give instructions, so I invented some and wrote them down here.

Before we start, you need to know what my system looks like:

CoCo 3, 512K
Hi-Res Mouse
20 Meg Hard disk (set up as "/d0")
35 Track floppy (set up as "/d1")
Parallel printer (set up as "/p1")
Tandy ACIAPAK (set up as "/t2")
WIZ ACIA driver (set up as "/m2w")
Magnavox RGB monitor

There is some other stuff, but it doesn't impact installation.

The first thing you should do is read the "Getting Started" section of the manual. Then read pages 9-28 through 9-31, which give you the format of MV's "environment file". The environment file configures MV to your system, and you will have to modify it in order to successfully install Multi-Vue.

Now, make a backup of the release disk(s). Tandy has distributed Multi-Vue on 2 sides of a single floppy disk, so you need to back up each side to a seperate "normal" floppy disk. Label the backups DISK #1 and DISK #2. Put the master in a safe place, and use ONLY the 2 backups for the rest of this procedure.

1) Make sure that all of the files from your Level 2 OS-9 release and CONFIG disks are on the hard drive. If they are not, you can put the OS-9 release disk in /d1 and enter:

```
OS9:chd /d1
OS9:dsave -s20 /d1 /d0 ! shell
```

Then put the CONFIG disk in /d1 and enter:

```
OS9:chd /d1
OS9:dsave -s20 /d1 /d0 ! shell
```

This takes a while, but does the trick. Remember, /d0 is the hard drive in the above example! If your hard drive has a different name, don't forget to change when typing.

2) Put DISK #2 in /d1, and copy everything from it to your hard drive:

```
OS9:chd /d1
OS9:dsave -s20 /d1 /d0 ! shell
```

3) Unless you want Multi-Vue to boot automatically, you must delete the AUTOEX file from your hard drive:

```
OS9:del /d0/cmds/autoex
```

It seems that this is one Tandy didn't tell us about. Any file named /d0/cmds/autoex will boot automatically whenever OS-9 is started.

4) Put DISK #1 in /d1, and copy the contents of its CMDS directory to the hard disk:

```
OS9:chd /d1/cmds
OS9:dsave -s20 /d1 /d0/cmds ! shell
```

5) Copy the environment file from DISK #1 to the hard drive:

```
OS9:copy /d1/sys/env.file /d0/sys/env.file
```

6) Edit the environment file to tailor it to your system. I had to change the following entries:

```
OLD FIELD       NEW FIELD
MONTYPE=0       MONTYPE=1
 Select RGB monitor
RAM=128         *RAM=128
 Comment out 128K code
*RAM=512        RAM=512
 Use 512K code
PTRRES=0        PTRRES=1
 Hi-Res mouse
RBFDEV=         RBFDEV=
    /d0,/d1     /d0,/d1,/d2,/d3,/h1,/dd
 Register all RBF devices
SCFDEV=         SCFDEV=
    /p,/t1      /p1,/t2,/m2w
 Register all SCF devices
PRPORT=/p       PRPORT=/p1
 Use my printer driver
```

7) Add the following line to your startup file:

```
merge /d0/sys/stdfonts /d0/sys/stdpnts_4 /
do/sys/stdptrs
```

8) Build a short command file to start Multi-Vue:

```
OS9:build /d0/xmv
tmode -echo
control -e
gshell &
tmode .1 echo
```

The CONTROL program uses the environment file to set up some of Multi-Vue's parameters, etc. GSHELL is the new graphic shell that is supplied with Multi-Vue: it will automatically create a graphics window for itself any time you run it.

9) Make a new OS9 boot disk that supports Multi-Vue. This will be identical to your existing boot file, with the following exceptions:

i) Use WINDINT.IO instead of GRFINT.IO

ii) Include W7.WD through W12.WD, to provide additional windows. NOTE: if you have 512K, you can also include W13.DD, W14.DD, and W15.DD

In this case, the easiest way to make a new boot disk is to use OS9GEN. If you have added anything to the standard Tandy boot, you probably already have a BOOTLIST file around somewhere — just add the new modules to it. For the standard Tandy boot, you can use the file "/d0/modules/bootlist.mv". In any case, you must put a new disk in /d1 and enter:

```
OS9:format /d1 r " "
OS9:os9gen /d1 #40K </d0/modules/
bootlist.mv
```

10) Users of XT-ROM may wish to install the new boot file on the hard disk. To do this, enter:

```
OS9:bootport /d1 /d0
```

11) Take a break. After all this you deserve it.

12) Boot the new system, either from the hard disk or the floppy disk.

13) THIS IS IT! Run Multi-Vue by entering:

OS9:xmv

Now hit the CLEAR key until you end up in the Multi-Vue GSHELL window.

14) Each time you execute the XMV file, another Multi-Vue graphic shell will be spawned. You can rotate between all of your spawned SHELL and GSHELL windows by depressing the CLEAR key.

A Brief Review of Multi-View

Summary: I give it a 5 (out of a possible 10). Impressive, as long as you didn't expect a Macintosh (which I did...). This program makes me want to go off and write a set of add-ins to make it what I want it to be.

The screens and icons look great. I wish that you could have MV come up in HI RES screen mode. As it is, you have to bring it up in LO RES and change over.

Mouse operation is very smooth, and the default mouse speed is very comfortable (hi-res mouse).

The documentation looks very good. There is an extensive section on accessing the windows from "C".

You cannot drag icons around on the screen, and there is no TRASH icon (Apple computer has a copyright on the trash can anyway). All file manipulation is done by typing in file names from the keyboard.

You cannot "double click" to execute an application.

There is no good way to open multiple directories on the desktop. You can only have one directory open at a time. Of course, since you can't drag icons around, much of the reasoning behind having multiple directories open does not apply.

GSHELL is written in "C", and it takes up about 16K (without any merged utilities). The standard OS-9 shell is written in assembler and uses about 1.5 K. This is a very big difference, and means you may not be able to run big applications (like the "C" compiler) from within GSHELL.

Several nifty "pop-up" utilities are provided with Multi-Vue. These include a clock, a calendar, a decimal and HEX calculator, and a method for escaping to the standard OS-9 shell.

The clock utility draws an analog clock face on the screen and updates it in real time! The clock face is oblong, rather than round, and the proportions of the hands make the clock difficult to read — especially in HI RES mode.

The calculator is very handy, especailly the HEX feature (for programmers like me). The screen definitions of the key boundaries are off a little bit; for example, the "=" key often registers as "3". Also, you must press the keys with the mouse kind of slooooowly if you want them all to register.

Chris Burke

Another angle...

Having just (finally!) purchased a hard disk system for my CoCo 3, I busily began to copy my 20 or 30 floppy disks to the new hard drive. Although I rarely use Multi-Vue, having it there and WORKING is a must. I heard rumors that there were problems running on a hard disk and sure enough the little bugs jumped right out and bit me. The following is a discussion of the changes to make to make your MV hard disk system build go smoothly.

Some Background

The diskette version of Multi-Vue is designed to run on /D0. /D0 is, unfortunately, hard-coded in "multistart". Since "autoex" is just a renamed copy of "multistart", the same problem applies. If you want MV to come up at boot time on your hard disk system then you will want to patch "autoex". If you want to start MV manually then delete "autoex" from /H0/CMDS and apply the patch only to "multistart". It is recommended that you patch both files if you put them on your hard disk.

Making Patches

This is the difference output from the patched and original versions. #1 is the original and #2 is the patched version:

```
Differences
byte        #1  #2
00000225    30  64
00000239    30  64
00000259    30  64
```

(change the "0" in "d0" to "d" to make "dd")

```
000004E7   AA 2A  (CRC byte 1)
000004E8    26 7D  (CRC byte 2)
000004E9   2D B0  (CRC byte 3)
Bytes compared:  000004EA
Bytes different:  00000006
```

Here is the procedure to patch "multistart" and "autoex". Note that it is assumed that "/dd" points to your hard disk. If /dd is not pointing to the hard disk then substitute the proper device name in place of "/dd" (e.g. /h0).

```
load multistart  (or autoex)
del -x multistart autoex  (this deletes the file from CMDS directory)
modpatch
l multistart  (link to multistart)
c 0225 30 64   (change "d0" to "dd")
c 0239 30 64                "
c 0259 30 64                "
v     (force the CRC to be correct)
<CTRL-BREAK>  (exit "modpatch")

save /dd/cmds/multistart multistart  (save the patched modules)
save /dd/cmds/autoex multistart
unlink multistart  (remove multistart from memory)
```

Now you must make the environment and setup changes in "SYS/env.file" and "startup". Use your favorite editor to make the following changes:

```
-in SYS/env.file -
RBFDEV=/h0,/d0,/d1,/d2 (edit this line to add your hard disk)
SCFDEV=/p,/t1
MONTYPE=1
*RAM=128
RAM=512
EXEC=/dd/CMDS (change "/d0" to "/dd" here)
*PROGRAM=Shell
*PARAM=i=/term
PALET15=0,3,3
-in startup -
montype r
* Echo welcome message
echo * Welcome to OS-9 LEVEL 2 *
echo * on the Color Computer 3 *
* Lock shell and std utils into memory
link shell
* set time from RTC
clkget -d &
echo
* ENTER THE NEXT LINE IN YOUR STARTUP FILE FOR MULTIVIEW
merge /dd/sys/stdfonts /dd/sys/stdpats_4 /dd/sys/stdptrs
```

Now you should reboot your system. If you use autoex, then MV should pop up running normally. If using "multistart" to manually begin MV then enter "multistart" at the OS9: prompt. MV should again pop up and run correctly.

Jim Johnson

In the next issue we will discuss Chris Dekker's "CoCoTop", an alternative toMulti-Vue. If you have ever used "X-Tree Gold" or any of the Norton file utilities on a PC, you will definately like CoCoTop...

Basic09 In Easy Steps

Chris Dekker

The last of the series... finishing up with windows

Did you try the program I gave you last time? If so, you must have seen that the last 20 or so characters on every line are left blank. We can use this space to print the characters associated with the codes in a line. You can do this by adding a few lines of code to the WHILE/ ENDWHILE loop:

```
PRINT "  ";
FOR i=0 TO 15
 char=PEEK(pointer+i)
 IF char<32 THEN PRINT ".";
 ELSE PRINT CHR$(char);
 ENDIF
NEXT i \PRINT
```

Adding this code gives you the core of a professional looking hexdump program. A few notes concerning the code: The last PRINT statement replaces the one at the end of the original loop. We print the codes 0-31 as dots because they can be interpreted as as control codes/sequences. This would really mess up your screen or even crash the program. Finally don't forget to DIMension "char" as a BYTE.

Back to our windows. If you (have) run the program I gave you, you can see windows at work. For this you need two device windows: one to run "dump", the other to make changes to the window. In the (unlikely) event you have never set up a second window; here is a fast way to get one. After you boot up and the OS9: prompt appears, type:

```
shell i=/w7&<ENTER>
```

press the CLEAR key to move to w7

```
display 1b31 1 0<ENTER>
```

background becomes black

```
display 1b31 8 3f<ENTER>
```

use white letters

Now you can use one window to experiment with: change colors, open and close overlay windows, change the working area, etc. If you run the "dump" program repeatedly in the other window, you can see the changes OS9 makes to the window descriptors.

Since the first 4K of (MMU) block 0 holds next to no information on windows, you can insert the following line just before the WHILE statement:

```
pointer=pointer+4096
```

The address that will now be tagged as 0000 by "dump" is the base address for the windows system's workspace. The first two pages (256 byte blocks) shown are used as "direct page" by GrfInt/WindInt and GrfDrv respectively. Direct page is a term related to one of the 6809's addressing modes. The blocks' main use is as scratchpad, although sometimes variables get passed along in these blocks too.

Some of the easier to recognize items in a window's descriptor are: starting coordinates (bytes 6 and 7), it's size (8/9), cursor coordinates (15/19) and working area (44/48). Note that for gfx windows the working area is expressed in pixels. You won't find the starting coordinates for the working area: that point can be found in the "starting address"; a pointer in bytes 4 and 5.

Do we have a use for knowing all this? Perhaps. It mostly depends on the program you write. For instance if you want to write a real slick wordprocessor, you may want to have the screen memory coincide with your program's text buffer. Any changes you make will immediately become visible on the screen.

Another use would be for graphics programs that want to load or save a screen real fast: map the screen in your addressing space and use Read/Write system calls. (A 6309 will save a 32K screen to ramdisk in less than a second with this technique.) And then there are cases where system calls don't work properly or don't exist, like for reading the cursor coordinates. Using the technique shown in "dump" you can read these values at the source.

Before we go on, however, I want to stress one point. Unless you are absolutely sure what you are doing, never ever write (POKE) values into block 0. The system call we used ($4F) does not make a copy of block 0, but maps the actual block into your addressing space. Any changes you make will immediately be available to the operating system. This can have dire consequences as I found out while writing 6309 screen drivers: a graphics program that ran fine on a 192 line screen kept crashing on a 200 line screen. Once it even damaged a floppy disk by overwriting trackdata.

The problem turned out to be the algorithm (in GrfDrv) that calculates scale factors. It calculated the wrong scaling factor for 200 line screens, which greatly upset the

operating system. However, all of this is little comfort if you just crashed your hard drive.

So, let's assume you are a careful person and want to read the cursor position on your window. How do you go about it? Especially if there are 4 or 5 window descriptors in the system. Which one do you have to read??

OS9 faces, of course, the same problem: Which window do I access? Fortunately for us it has also solved that problem. The second direct page (mentioned above) contains two pointers: one to a window descriptor and the other to a windowtable entry. These two pointers are updated by the SELECT function (As in RUN gfx2("select")). This function is also executed when you press the CLEAR key. So the pointers always point to the window you are looking at.

Now we are home free! All you do is make sure the correct window is displayed, read the pointer, calculate the descriptor's starting address and read the appropriate offsets. The code to do so looks like this:

```
RUN gfx2(path,"select")
regs.x=2 \RUN syscall($0A,regs)
```

The above code makes sure we will read the correct pointers. If you want to read data about the screen you are looking at: set "path" to 1. Do not omit the second line or your program will eventually crash. This instruction puts your application to sleep for 1/30 of a second allowing OS9 and the hardware to catch up to it.

```
regs.x=0 \regs.b=1
RUN syscall($4F,regs)
\(* map block 0
 IF LAND(regs.cc,1)=1 THEN ERROR regs.b \ENDIF
```

```
    pointer=regs.u+$112E
  \(* address of pointer
   pointer=256*PEEK(pointer)+PEEK(pointer+1)
    pointer=pointer+regs.u
  \(* points to windowdescriptor
   xcor=256*PEEK(pointer+14)+PEEK(pointer+15)
    ycor=PEEK(pointer+17)
    RUN syscall($50,regs)
  \(* release block 0
```

Xcor and ycor now hold the current cursor position. On graphics screens they will be expressed in pixels rather than character positions. If your program needs to read the coordinates repeatedly AND it has enough room in it's addressing space; then you can speed things up by releasing block 0 when your program exits. This way it can read the coordinates anytime it wants without additional overhead.

Suppose you don't want to read cursor coordinates, but access the screen directly with your program for loading or saving it, animation, anything you want. This can be accomplished too. The start of that code looks a lot like the code above: before the line xcor=..., there is only one change. You must replace the term $112E with $1130. This directs your pointer to the windowtable.

Of course we can't read the cursor position now, so the lines xcor=..., ycor=... get replaced by the following code:

```
    Wtype=PEEK(pointer)
    Wblock=PEEK(pointer+1)
   Wstart=256*PEEK(pointer+2)+PEEK(pointer+3)-
$8000
    screenwidth=PEEK(pointer+4)
```

Wtype stands for window type. OS9 internally uses a
different format than the codes we use in the DWSET command.

So we have to do a little conversion:

```
    IF LAND(Wtype,128)=128 THEN
      Wtype=LAND(Wtype,3)
  \(* text screen, code 1/2
    ELSE Wtype=Wtype+4
  \(* gfx screen, code 5/8
    ENDIF
```

Wblock is the number we're after. This is the (starting) block that holds the actual screen data. In the case of gfx screens (16K or 32K) this is the starting block number. Screens and buffers, though, always get allocated as contiguous memory. So if "Wblock" shows block 50 as starting block for a 32K screen, you can be sure that blocks 51, 52 and 53 also belong to that screen.

This contiguous memory thing, by the way, can also cause the computer to hang up if memory becomes too fragmented. This happens, for instance, when you want to open a 16K overlay window and there are only 8K blocks available. It would be nice if we got an error but alas...

Wstart holds the address of the first pixel or character of a window. Overlay and text windows do not necessarily start at byte 0 in the block so it would be wise to use Wstart in your calculations. The term - $8000 has to do with the contents of the CoCo's DAT registers, which is more or less outside the scope of the article.

Screenwidth is the number of bytes needed for one screenline. This would be 80 (40 column screens) or 160 (80 column screens). Note that this number not necessarily reflects the width of a window on the screen.

Armed with all this knowledge, you should have no problems accessing the screen through the $4F/$50 pair of system calls. Just remember this: hang on to the value returned in register U by system call $4F. This becomes the base address of your screen. If you do not add this value to any address your program might calculate, the program will sooner or later overwrite it's own data. I doubt you will like the results of that.

My final remarks concern the structure of the CoCo's text screens. Every character (including spaces) you see on the screen is represented by 2 bytes in memory The first byte holds the character code. Th. second byte holds the character's attributes. It actually holds 4 variables (assuming you stored the value of this byte in the variable "attr"). Here is what they mean:

if LAND(attr,128)=128 then character blinks

if LAND(attr,64)=64 then character is underlined

```
background=LAND(attr,7)
foreground=LAND(attr,56)/8
```

Background and foreground return register numbers. The colors they display are determined by values in the window's windowtable entry (see previous article).

The block cursor on text screens is formed by displaying a space in reverse video. On text screens reverse video is achieved by switching the positions of fore- and background registers in the attributes byte. Changing the register numbers will (most likely) change the colors of the character displayed.

EDTASM6309 v2.02 - $35.00
Patches Tandy's Disk EDTASM to support Hitachi 6309 codes! Supports all CoCo models, including stock 6809 models. CoCo 3 version uses 80 column screen, runs at 2MHz. YOU MUST HAVE A COPY OF DISK EDTASM. This is a PATCH ONLY! It will not work with "disk patched" cartridge EDTASM

CC3FAX - $35.00
Receive and print weather fascimile maps from shortwave! The US weather service sends them all the time! Requires 512K CoCo3 and shortwave receiver. Instructions for simple cable included.

HRSDOS - $25.00
Move programs and data between DECB and OS-9 disks! Supports RGB-DOS - move files easily between DECB and OS-9 partitions! No modifications to OS-9 modules required.

DECB SmartWatch Drivers - $20.00
Access your SmartWatch from DECB! Adds function to BASIC (DATE$) for accessing date and time. *Only $15.00 with any other purchase!*

RGBOOST - $15.00
Make the most of your Hitachi 6309 by using it with DECB! Uses 6309 functions for up to 15% gain in speed. Compatible with all programs tested to date! *Only $10 with any other purchase!*

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
Please add $4 S&H per order

Northern Xp[illegible]r

CoCo OS-9:
NitrOS9 - Upgrade OS-9 Level II for the CoCo to a new level of performance! Requires Hitachi 6309. Call or write for upgrade info. $29.95
Shanghai:OS-9 or **Thexder:OS-9**
Send manual or ROM Pak to prove ownership. Transfers and modifies code to run under OS-9! $29.95
Rusty - Launch DECB programs from OS-9! $20.00
SCSISYS 2.2 - SCSI hard drive drivers... fast! Supports 256 or 512 byte sector drives! $25.00
Hitachi 6309 CPU $15.00
SIMM 512K Memory Upgrade - runs cooler!
W/512K - $44.95 w/o memory - $39.95

OS-9/68000:
OSTerm 68K - v2.2terminal program. TTY/ANSI/VT100/KWindows support $50.00

7 Greensboro Cres
Ottawa, ON K1T 1W6
CANADA

All prices in US funds. Check or Money Order only.
Prices include S&H.

for all your CoCo hardware needs, connect with

CoNect

449 South 90th Street
Milwaukee, WI 53214
(pulland@omnifest.uwm.edu)

That thing that Tandy calls a serial port on the CoCo has always been a problem. It was designed with minimal cost in mind, and never upgraded. Even Tandy tried to fix it with their RS-232 Pak, but even it was only half done! Our Fast 232 port uses a 16 byte buffer at alleviate missed characters at any speed and also has ALL RS-232 lines implemented. It is easy to set up with jumpers for different addresses. A daughterboard can be purchased to easily add a second fast serial port! And all this in a cartridge the size of a ROM Pak! 6809 and 6309 OS-9 drivers included. Completely supports up to 57,600 bps, limited support for 115,000 bps.

Fast 232 - $79.95
Daughter Board - $45.00

Check with us for complete disk drive systems, misc. hardware items, hardware repairs, and hard to find new and used CoCo software!

ADVERTISER'S INDEX

What are you waiting for?

Get your friends to subscribe to the only magazine that still supports the Tandy Color Computer... *"the world of 68' micros"*!
The more people who want the support, the longer it will be here!

Wittman Computer Products

Now all new ways to contact us!

Our Web Page:

http://www.frontiernet.net/~wittman/wcp.html

New e-mail address:

wittman@frontiernet.net

24 hour Fax line:

(716) 494 - 2875

Voice mail 8 am to 10 pm EST:

(716) 494 - 1506

Or U.S. Postal Service:

Wittman Computer Products
Wm. L. Wittman
39 South Lake Avenue
Bergen, NY 14416

New Catalog available: October 1, 1996

All NEW and updated.